Borut Jereb,* Ljubo Pipan,** Aleš Klofutar
Institut J. Stefan
* Gorenje raziskave in razvoj, T. Velenje
** Fakulteta za elektrotehniko, Ljubljna

Descriptors: TRANSPUTERS, PARALLEL PROCESSING

Za doseganje večje zmogljivosti računalnikov, je znanih veliko teoretičnih pristopov. Nekatere od njih je možno realisirati, nekatere delno, nekateri pristopi pa so s sedaj poznanimi tehnologijami nerealni. Pri transputerjih je načrtovalcem uspelo realisirati nekaj zanimivih zamisli. Članek je pregleden in obravnava predvsem najnovejši transputer T800, ki je relativno cenen in zmogljiv gradnik v eno ali večprocesorskem sistemu. Pri večprocesorskem sistemu so lahko transputerji močno ali šibko povezani. Model T800 ima obenem veliko numerično moč. Ta moč je posledica nekaterih izvirnih rešitev, ki so opisane v članku.

TRANSPUTERS

Several theoretical principles for the achievement of greater computer capability already exist. Some of these can be carried out, some are only partially aplicable, while many theories can not yet be realised, given today's technology. Since experts have succesed in discovering several good and applicable ideas and possibilities in transputer designing, this paper is meant to give the reader a lucid survey of transputers, with special emphasis placed on the newest model: T800. This is a relatively cheap and capable unit, designed for use in both single or multiprocessor systems. In the latter, the transputers can be either tightly or loosely linked and the T800 model has, at the same time, great numerical power - a result of the several original solutions presented further on in this paper.

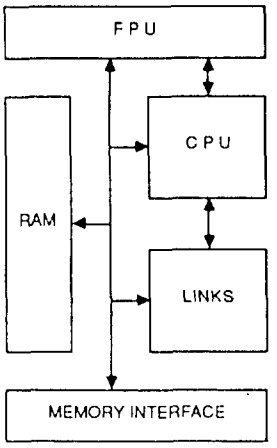
0 UVOD

Članek opisuje družino transputerjev. To so izdelki podjetja Inmos, ki se od klasičnih mikroprocesorjev razlikujejo po prilagojenosti paralelnemu procesiranju.

Prva štiri poglavja podajajo splošen opis družine transputerjev. Peto poglavje na kratko opisuje nekatere zmožnosti transputerja IMS T800 (predvsem enoto za delo s števili v plavajoči vejici). Sledi še zaključek.

1 KONCEPT IN ARHITEKTURA TRANSPUTERJA

VLSI tehnologija omogoča ceneno izdelavo velikega števila enakih integriranih vezij velike zmogljivosti. Z uporabo več enakih integriranih vezij lahko realiziramo sistem z neko mero paralelizma oz. sočasnosti.



slika 1
Bločni diagram transputerja IMS T800

Transputer je VLSI integrirano vezje, ki vsebuje: procesor (CPU), pomnilnik (RAM), komunikacijske kanale (LINKS) za direktno povezavo z ostalimi transputerji v tako imenovani mreži transputerjev in vmesnik za zunanji pomnilnik (MEMORY INTERFACE); nekateri, odvisno od tipa, tudi enoto za delo s števili v plavajoči vejici (FPU). (slika 1).

Načrtovalcem tega vezja je uspelo napraviti vezje, ki je dobro prilagojeno paralelnemu procesiranju. Dodatek k paralelizmu realiziranem z večimi transputerji je velika mera notranjega paralelizma v samem transputerju. Samostojen sistem lahko predstavlja že en sam transputer ali pa več v mrežo povezanih transputerjev. Kot primer slika 1 podaja bločni diagram transputerja IMS T800.

1.1 PROCESOR IN POMNILNIK NA SKUPNEM VEZJU

V sistemih sestavljenih iz več VLSI vezij (procesor, pomnilnik itd), je hitrost prenosa podatkov med vezji, glede na hitrost delovanja samih vezij, zelo majhna. Obenem pa vsaka operacija, ki jo izvede procesor, zahteva uporabo pomnilnika. Zaradi slednjega sta pri transputerjih procesor in pomnilnik sestavna dela enega samega vezja. Tipična velikost pomnilnika na vezju je pri sedanjih transputerjih nekaj Kslogov.

1.2 KOMUNIKACIJE MED TRANSPUTERJI

Povezave med vezji in dodatna vezja za upravljanje povezav pomenijo glavno omejitev pri zmanševanju velikosti celotnega sistema sestavljenega iz integriranih vezij. Iz omenjenega sledi, da želimo zmanjšati število povezav med integriranimi vezji in obenem to povezavo čim bolj poenostaviti oz. uporabiti čim manj dodatnih vezij za upravljanje povezav. Transputer lahko povežemo z ostalimi transputerji s serijskimi enosmernimi "point-to-point" povezavami. Pri tem ne potrebujemo nikakršnih dodatnih vezij.

S tem zmanjšamo število povezav med integriranimi vezji in zelo poenostavimo upravljanje komunikacij.

1.3 ENOSTAVEN PROCESOR Z MIKROKODIRANIM RAZPOREJEVALCEM OPRAVIL

Transputer vsebuje enostaven sekvenčni procesor z majhnim številom instrukcij. Dodani sta specializirani skupini instrukcij za aritmetične operacije v plavajoči vejici in za razporejanje opravil.

Proces, ki se izvaja v transputerju lahko, vsebuje večje število sočasnih procesov, katerih sočasnost transputer podpira interno. To je omogočeno z mikrokodiranim razporejevalcem opravil, ki odmerja procesorjev čas sočasnim procesom. Razporejevalec omogoča dva prioritetna nivoja.

1.4 TIPI TRANSPUTERJEV

IMS T414 je 32 bitni procesor s pomnilnikom dveh Kilogov na vezju. Ima 32 bitno povezavo s zunanjim pomnilnikom in štiri kanale za povezavo z ostalimi transputerji. Hitrejša verzija tega izdelka IMS T414-20 omogoča tipično 10 MIPS.

IMS T212 je zelo podoben pravkar opisanemu tipu. Razlika je v širini podatkovnih poti. Povezava z zunanjim pomnilnikom je 16 bitna.

Obstajajo še procesorji serije T212 za kontrolo diskovnih in disketnih pogonov in procesorji, ki omogočajo priključevanje transputerskih sistemov na običajne sisteme preko vodila.

IMS T800 je najnovejši izdelek Inmosa. To je 32 bitni procesor. Na vezju so štirje Kilogi pomnilnika in enota za računanje s števili, katerih zapis je po ANSI/IEEE Standardu [6]. Izvedba IMS T800-30 omogoča 2.5 MFLOPS. Pri tem vezju je ohranjena kompatibilnost z vezjem T414-20 na nivoju nožic vezja[1].

2 PROGRAMIRANJE

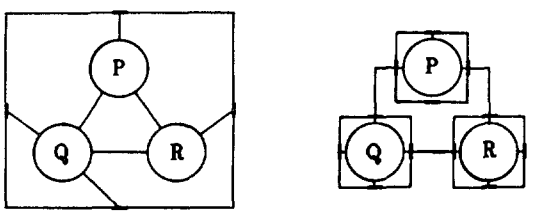
Če je osnovni gradnik paralelnega sistema transputer, potem nam proces predstavlja osnovni programski gradnik. Sistem transputerjev lahko načrtujemo in programiramo z jezikom, ki ga ponuja proizvajalec transputerjev - to je z jezikom Occam. Zaenkrat je poleg Occama na voljo še nekaj visokonivojskih jezikov. To so C, Fortran, Pascal in Fifth (podoben programskemu jeziku Forth) [1,6,8,9,10].

Program pisan v enem od visokonivojskih jezikov in nato preveden predstavlja modul. Z occam modulom nato povezujemo in konfiguriramo module neodvisno od originalnega jezika, v katerem je bil nek modul napisan, pred prevajanjem v kodo.

Pri izvajanju programa lahko pride do nekaterih usodnih napak (aritmetične prekoračitve, prekoračitve pri indeksiranju polj, deljenje z nič...), ki ustavijo procesor. Ko se pojavi takšna napaka, se postavi zastavica za napako in napaka se lahko obdela interno z programsko opremo ali pa se obdela zunaj z primerno strojno opremo (npr sosednji transputer v mreži transputerjev). V slednjem primeru zunanja logika zazna posebno stanje, v katerem se nahaja procesor na posebni nožici integriranega vezja.

2.1 OSNOVNI KONCEPT PROGRAMSKEGA JEZIKA OCCAM

Programski jezik Occam so razvili za programiranje sočasnih distribuiranih sistemov. Poudarek je na besedi distribuirani, saj dosedanji jeziki praviloma niso podpirali distribuiranih sistemov in temu ekvivalentnega razmišljanja.



Program na enem transputerju Program na treh transputerjih

slika 2
Realizacija procesov pri transputerjih

Razvoj jezika Occam je potekal sočasno z razvojem transputerja in ga nekateri jemljejo kot zbirni jezik za transputerje. Ima zelo malo (32 besed) rezerviranih besed, kar daje slutiti veliko prilagojenost jezika sami arhitekturi transputerja.

Izvajanje procesa v Occamu, je formalno ekvivalentno izvajanju programa na transputerju. Tako postane proces materialni in programski gradnik večprocesorskega sistema. Sočasni procesi so realizirani na mreži transputerjev; komunikacija med njimi in z zunanjimi napravami pa je realizirana s kanali. Konfiguracijo programa na mreži omogoča konstrukt PLACE PAR (pojem konstrukta je razložen kasneje). Occamov program je možno izvajati tudi na enem procesorju, ki deli čas med 'sočasne' procese. Glej sliko 2. Pri tem konfiguracija in število transputerjev nimata nikakršnega vpliva na logično obnašanje programa.

Sintaksa Occama uporablja zamikanje od levega roba za nazačevanje programske strukture. Vsak proces in vsak konstrukt je predstavljen z vrstico v programu.

2.1.1 PROCESI

Po zagonu procesa le ta izvaja akcije in se nato ustavi ali pa je ustavljen. Program pisan v Occamu je sestavljen iz treh osnovnih procesov. Ti so: prireditveni, vhodni in izhodni proces. Njihov zapis je naslednji:

v := e	prireditev izraza e spremenljivki v
c ! e	izhod izraza e v kanal c
c ? v	vhod spremenljivke v iz kanala c

Vsak Occamov kanal omogoča komunikacijsko pot med dvema sočasnima procesoma. Komunikacija je sinhronizirana in se izvede, ko sta oba - to je oddajajoči in sprejemajoči proces - pripravljena. Po končanem prenosu podatkov se oba procesa nadaljujeta.

2.1.2 KONSTRUKTI

Osnovne procese kombiniramo v konstrukte. Konstrukt je sam zase proces in ga lahko uporabimo kot del naslednjega konstrukta. Osnovna značilnost konstruktov je, da se začnejo s karakteristično oznaćbo, ki ji v naslednji vrstici - z zamikom glede na zgornjo - sledi lista osnovnih procesov ali/in konstruktov. Osnovni konstrukti so:

SEQuential	procesi se izvajajo en za drugim
PARallel	procesi se izvajajo sočasno
IF	proces se izvede ob izpolnjenosti pogoja
ALTernative	prvi pripravljen proces se izvede

Pri sekvenčnem izvajanju, se komponente procesov izvajajo ena za drugo. Sekvenčni konstrukt se konča s koncem izvajanja zadnje komponente konstrukta.

Komponente paralelnega konstrukta se izvajajo sočasno. Vsaka komponenta procesa operira na svojih spremenljivkah in komunicira z ostalimi sočasno delujočimi procesi preko kanalov. Paralelni konstrukt se konča le, če so se končale vse komponente konstrukta.

Tako je naslednji program sestavljen iz dveh paralelnih procesov. Prvi proces bo sprejel spremenljivko next.problem iz kanala source. Drugi proces pa je sestavljen iz dveh procesov, ki se bosta izvedla zaporedno: prvi računa computr.next.solution, drugi pa po zaključku prvega pošlje solution v kanal result.

```
PAR
    source ? next.problem
    SEQ
        computr.next.solution (this.problem, solution)
        result ! solution
```

Klasične sekvenčne programe lahko v Occamu napišemo tako, da uporabimo spremenljivke in prireditve v sekvenčnih konstruktih.

Pri pogojnem konstruktu se testirajo komponente zaporedno. če je komponenta pravilna se izvrši odgovarjujoči proces. Vedno se izvrši le en proces. Naslednji primer prikazuje uporabo konstrukta IF pri primerjavi števil a in b.

```
IF
    a > b
        order := gt
    a < b
        order := lt
    TRUE
        order := eq
```

Alternativni konstrukt omogoča večim procesom hkratno pripravljenost za sprejem podatkov iz kateregakoli od možnih kanalov. Sprejem se bo izvršil najprej iz tistega kanala, ki ga prvega uporabi nek drug proces za izhod. Pri tem imamo možnost dela z zunanjimi in notranjimi dogodki. V klasičnih mikroprocesorjih se takšne stvari rešujejo na nivoju sbirnika s prekinitvami. Za primer si oglejmo primer, kjer čakamo na signal na kanalu count in total. Če pride signal na kanal count povečamo spremenljivko counter za ena, v drugem primeru pa pošljemo skozi kanal out vrednost counter, ki jo nato še postavimo na nič.

```
ALT
    count ? signal
        counter := counter + 1
    total ? signal
        SEQ
            out ! counter
            counter := 0
```

Tudi ponavljanje je uporabljeno kot konstrukt. V spodnjem primeru se proces P izvršuje dokler ni pogoj condition napačen (false). Primer:

```
WHILE condition
    proces
```

Uporaba konstrukta kopiranja je razvidna iz naslednjega primera:

```
SEQ i = base FOR count
    a[i] := i
```

Slednje je ekvivalentno zapisu:

```
SEQ
    a[base]              := base
    a[base + 1]          := base + 1
    .....
    a[base + count - 1]  := base + count - 1
```

V običajnih jezikih uporabljamo za takšno nalogo ukaz FOR.

2.1.3 TIPI

Sedanja verzija Occama podpira več podatkovnih tipov, kakor tudi večdimensionalna polja. (Tipi:CHAN OF type (kanal tipa ...), TIMER, BOLL, BYTE, INT, INT16, INT32, INT64, REAL32, REAL64, [n,m,...] type). Polja se lahko prirejajo, prenašajo med procesi in uporabljajo kot parametri v procedurah. Occam obenem omogoča, da del polja obravnavamo kot polje. Za primer si oglejmo naslednji program, ki deklarira celoštevilčno polje desetih elementov s imenom a. Vrednosti elementov zajema paralelno iz kanalov c in d (prvih pet elementov iz kanala c, drugih pet pa iz kanala d).

```
[10] INT a
PAR
    c ? [a FROM 0 FOR 5]
    d ? [a FROM 5 FOR 5]
```

2.1.4 PROCEDURE, IZRAZI, ČASOVNIK IN ZUNANJE ENOTE

Procedura je proces, ki mu lahko damo ime. Npr:

```
PROC square (INT n)
    sqrt := n*n
```

Izrazi so sestavljeni iz operatorjev, ki jih najdemo v tabeli 2, iz spremenljivk, števil, logičnih izrazov ter predklepaja in zaklepaja.

+, -, *, /, REM	integer, real
PLUS, MINUS, TIMES, AFTER	integer
=, <>	enostaven iz.
>, <, ≤, ≥	integer, real
AND, OR, NOT	boolean
∧ (bitwise and), ∨ (bitwise or)	
>< (bitwise xor), ~ (bitwise not)	integers
<<, >> (premikanje)	integer

Tabela 1
Operatorji v Occamu

Vsak proces lahko ima svoj neodvisen časovnik, ki ga uporabi za svoje meritve ali za razdeljevanje dela v realnem času. Časovnik prebere vrednost v spremenljivko tipa INT. Npr:

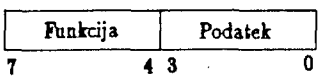
```
tim ? v
```

postavi spremenljivko v na trenutno vrednost prostotekoče ure, ki je deklarirana kot časovnik tim.

Dostop do zunanjih enot je v Occamu omogočena s mehanizmom vhodno/izhodnih vrat. Vrata se uporabljajo podobno kot kanali. Podobno lahko le en proces bere iz v/i vrat in le eden daje na v/i vrata.

3 KODIRANJE INSTRUKCIJ

Vsi transputerji imajo enak osnovni nabor maloštevilčnih instrukcij. Vsaka instrukcija vsebuje osem bitov, ki so razdeljeni v dve skupini po štiri bite. (Glej sliko 3.)



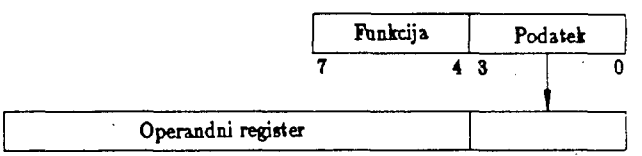
slika 3
Format transputerjeve instrukcije.

Pomembnejši štirje biti tvorijo funkcijsko kodo, preostali štirje so podatki. Dobimo 16 "direktnih" funkcij (nalaganje, shranjevanje, skoke, klice...).

Vse instrukcije se izvedejo tako, da se spodnji štirje biti prepišejo v spodnje štiri bite operandskega registra, katerega vsebina se kasneje uporabi kot operand instrukcije (glej sliko 4). Pri tem se vse instrukcije, razen Prefix instrukcij (njihova funkcija bo razložena kasneje), končajo s brisanjem operandskega registra. Tako je le ta pripravljen za naslednjo instrukcijo.

Ker pa takšno kodiranje dopušča samo štiri bitne operande, imamo med zgornjimi 16. funkcijami dve, ki omogočata razširjavo velikosti operandov. To sta Prefix in Negative Prefix. Prefix instrukcija napolni spodnje štiri bite operandskega registra s svojim podatkovnim poljem in potem premakne vsebino operandskega registra za štiri mesta v levo (šiftanje). Negative Prefix instrukcija je podobna pravkar opisani Prefix instrukciji, le da komplementira operandski register pred premaknitvijo vsebine za štiri mesta. Tako lahko operand s uporabo Prefix instrukcij povečujemo do velikosti operandskega registra.

Naslednja od zgornjih 16. funkcij je Operate, ki tretira svoj operandski del - osiroma vrednost operandskega registra - kot operacijo nad vrednostmi v registrih procesorja. Ta funkcija omogoča kodiranje še dodatnih 16. operacij v enem slogu.



slika 4
Polnjenje operandnega registra pri IMS T800.

S Prefix funkcijo lahko razširimo tudi operande funkcije Operate, kar je ekvivalentno povečanju nabora instrukcij. Seveda so instrukcije kodirane tako, da so najpogosteje uporabljane instrukcije predstavljene brez Prefix instrukcij. Merjenja so pokazala, da je okoli 70% instrukcij, ki se izvajajo, kodiranih v enem slogu. [3]

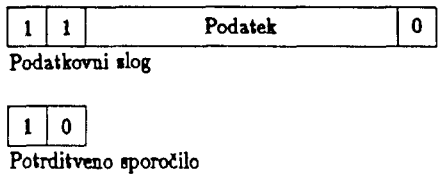
T800 ima dodatne instrukcije za delo s FPU. Pravtako vsebuje instrukcije za barvno grafiko, raspoznavanje vzorcev in implementacijo kod za odpravljanje napak. To je realizirano z zgoraj opisano možnostjo, ki omogoča razširitev nabora instrukcij. [5,7]

4 KOMUNIKACIJSKE POVEZAVE

Štiri identične dvosmerne povezave omogočajo sinhronizirano komunikacijo med procesorji in komunikacijo z zunanjim svetom. Vsaka povezava vsebuje vhodni in izhodni kanal. Povezava med dvema transputerjema je realizirana s povezovanjem vmesnika povezave enega transputerja na vmesnik povezave drugega transputerja. Vsak poslan podatkovni slog mora biti potrjen preko vhodnega kanala iste povezave. To je sinhronizacija, ki poteka na vseh štirih povezavah transputerja avtomatično in ne zahteva dodatnega programiranja.

Ko linija ni aktivna je izhodni kanal na nizkem nivoju. Vsak podatkovni slog se prenese kot zaporedje visokega start bita, enega visokega bita, tema bitoma sledi osem podatkovnih bitov in nizek stop bit. Potrditev, ki jo čaka oddajnik vsebuje visok in nizek bit (glej sliko 5) in je indikator za dvoje: proces je sprejel podatek in vmesnik je pripravljen za sprejem naslednjega sloga.

Pošiljanje potrditvenih paketov, preden se podatkovni paket popolnoma sprejme, poveča zmožnost povezav. IMS T414 nima implementiranega pravkar opisanega pošiljanja in dosega le 0.8 Mslogov na sekundo. Z implementacijo prekrivanja in zadostnih izravnalnikov za povezavo, je IMS T800 omogočena več kot dvakrat višja hitrost prenosa. [1,2,3,6,7]



Podatkovni slog

Potrditveno sporočilo

slika 5
Elementi komunikacijskega protokola

5 IMS T800

IMS T800 je naslednik T414, ki je bil prvi širše uporabljen predstavnik iz družine transputerjev. T800 vsebuje, za razliko s T414, integrirano enoto za delo s števili v plavajoči vejici (FPU). To predstavlja, glede na običajne rešitve s koprocesorji, le malo površino dodatnega silicija. Običajno zahteva zadovoljivo povečanje numerične zmogljivosti dodatno površino silicija, ki se giblje v razredu velikosti površine silicija, porabljene za izvedbo mikroprocesorja. Seveda to avtomatično zahteva eno ali več integriranih vezij z vso logiko, ki je potrebna za delovanje samega vezja (Weitek WTL1167 koprocesor za mikroprocesor Intel 80386 zahteva tri integrirana vezja). FPU deluje sočasno s centralno procesno enoto (CPU) in pod kontrolo CPU. (Glej sliko 1.)

5.1 ARHITEKTURA

Procesor transputerja T800 ima malo registrov, kar je kompenzirano s zelo hitrim pomnilnikom na vezju. Šest registrov in enostaven instrukcijski nabor omogoča enostavno kontrolno logiko in enostavne ter hitre podatkovne poti. Registri procesorja so: kazalec na delovno področje, kjer so shranjene lokalne spremenljivke; kazalec instrukcij, ki kaže na naslednjo instrukcijo, ki se naj izvrši; operandni register, kjer se nahaja operand instrukcije; registri A, B in C, ki predstavljajo strojno izveden sklad. Slednji trije registri se uporabljajo za aritmetiko pri naslavljanju, za celoštevilčno aritmetiko ter za logične operacije.

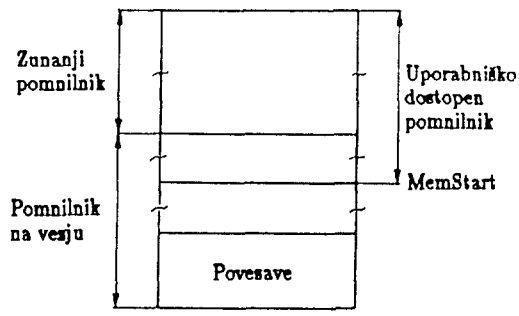
Tudi FPU ima tri registre (AF, BF in CF) v obliki sklada za računanje.

Naslovi za podatke, ki so zapisani v plavajoči vejici se formirajo na skladu CPU. Obenem je pod kontrolo CPU prenos vrednosti med naslovljenimi pomnilniškimi lokacijami in FPU. Ker je CPU sklad uporabljan le za naslavljanje vrednosti v plavajoči vejici, je dolžina besede CPU neodvisna od dolžine besede FPU. Tako dosežemo, da isti FPU na vezju T800 lahko uporabljata T212 (16 bitna beseda) in T414 (32 bitna beseda).

Registrski sklad FPU je podvojen. Pomembnost tega se vidi ob preklopu T800 v delovanje z visoko prioriteto, ne da bi bilo potrebno prepisovanje vsebine sklada v pomnilnik. Rezultat slednjega je zelo ugoden časovni odziv na prekinitve [3,6,7].

5.2 ORGANIZACIJA NASLOVNEGA POLJA

Celoten pomnilniški prostor je naslovljiv po slogih. Naslovi med #80000000 in #80000FFF naslavljajo pomnilnik na vezju (to je 4 Ksloge). Uporabniški pomnilnik se začenja na naslovu #80000070. Lokacija s tem naslovom je označena kot MemStart (Memory Start) [3]. Glej sliko 6. Naslovi povezav so v spodnjem delu pomnilnika na vezju.



slika 6
Organizacija naslovnega polja

5.3 INSTRUKCIJE ZA DELO S ŠTEVILI V PLAVAJOČI VEJICI

Jedro množice instrukcij za delo v plavajoči vejici so določili v fazi pred načrtovanjem IMS T800. To jedro vsebuje enostavne operacije vpisovanja in branja FP operandov ter osnovne aritmetične operacije. Po drugi strani pa je statistika, ki je bila izdelana na osnovi fortranskih programov pokazala, da bi z dodatkom nekaterih bolj kompleksnih instrukcij povečali učinkovitost in kompaktnost kode. Odločiti so se morali za najustreznejši nabor instrukcij. Zato so opravljali raziskave učinkovitosti predlaganih razširjav naborov instrukcij. Tak nabor so potem testirali v numerično orientiranih programih. Pri tem so za vsak predlagan nabor instrukcij skonstruirali prevajalnik, program prevedli z njim in tako dobljeno kodo testirali na simulatorju. V nadaljevanju sledi opis resultirajočega nabora instrukcij.

IMS T800 prenaša operande med pomnilnikom transputerja in skladom FPU z uporabo instrukcij za shranjevanje in nalaganje števil v plavajoči vejici. Obstajata dve skupini takšnih instrukcij: ena za števila enojne dolžine in ena za števila dvojne dolžine.

Naslov operandov v plavajoči vejici se izračuna na CPU skladu, nakar se operand naloži iz naslovljene pomnilniške lokacije na FPU sklad. Omogočena sta dva načina naslavljanja FP operandov: direktni in indirektni. Slednji način naslavljanja olajša delo s polji.

Operandi na FPU skladu imajo oznake, ki predstavljajo njihovo dolžino. Oznaka operanda se nastavi, ko operand naložimo oz. izračunamo. Te oznake zmanjšajo število instrukcij, ki jih rabimo pri aritmetiki v plavajoči vejici. Npr. ne rabimo instrukcije za seštevanje dolgih besed in za seštevanje kratkih besed, temveč preprosto le instrukcijo za seštevanje.

Operaciji sa branje resultatov iz FPU shranita prebrano vrednost iz FPU sklada v transputerjev pomnilnik. Za branje ne obstajajo indeksne instrukcije. To izgleda na prvi pogled presenetljivo, vendar izvira iz dejstva, da je v programih manj operacij branja iz FPU, kot pa vpisovanja v FPU. Zato indeksno naslavljanje pri branju ni realizirano.

Enojne instrukcije omogočajo najpogostejše operacije v FP: seštevanje, odštevanje, množenje, deljenje in primerjavo. Rezultat primerjave se prenese v register CPU.

Zaradi pogostega seštevanja in množenja v programih so v smislu čim večje kompaktnosti kode in hitrosti izvajanja nekatere instrukcije sestavili iz več osnovnih instrukcij. Npr instrukcija prištevanja operanda operandu na skladu je enakovredna dvema operacijama: vpisu operanda na sklad in seštevanju operandov na skladu. [3,6,7]

5.4 SOČASNE OPERACIJE FPU IN CPU

Pri IMS T800 dela FPU sočasno s CPU. Ta sočasnost zelo izboljša značilnosti v realnih problemih, kjer so elementi polja relativno težko dostopni. To je razvidno iz "Livermore Loops" testa, ki bo opisan v nadaljevanju. Ta test je množica majhnih jeder napravljenih tako, da predstavljajo možne tipe izračunov. Posebno vsebuje dostope do dvo in trodimensionalnih polj, to je tam kjer transputerjeva sočasnost pokaže zelo dobre resultate. Prevajalnik izbere najugodnejši vrstni red računanja naslovov in s tem poveča časovno prekrivanje.

Pri testu "Livermore Loops" je IMS T800-30 dosegel 2.25 MFLOPS, IMS T800-20 1.5 MFLOPS, T414-20 0.09 MFLOPS in VAX 11/780 (s PF pospeševalnikom) 0.54 MFLOPS. Program napisan v Occamu za ta test ima obliko [7]:

```
- LIVERMORE LOOP 7
SEQ k = 0 FOR n
    x[k] := u[k] + ((( r*(z[k] + (r*y[k]))) +
                    (t*((u[k+3] + (r*(u[k+2] + (r*u[k+1])))))))) +
                    (t*((u[k+6] + (r*(u[k+5] + (r*u[k+4])))))))
```

5.5 ZMOŽNOST IMS T800 ZA FP OPERACIJE

Časi izvajanj FP operacij niso zanesljivo merilo hitrosti izvajanja pravih numerično orientiranih programov. Zaradi tega primerjamo numerično učinkovitost procesorjev s Whetstonovim preiskusom. To je program, ki je dobra imitacija znanstveno-tehničnega programa. Vsebuje ustrezno število in strukturo operacij v plavajoči vejici, klicov procedur, indeksiranja polj in računanja transcendentnih funkcij.

Tabela 2 podaja zmožnosti IMS T414 in IMS T800 v primerjavi s ostalimi procesorji glede na Whetstonov test. IMS T414 je trikrat počasnejši kot koprocesor MC68881, vendar ima kombinacija MC68000/MC68881 le 25% večje zmožnosti kot T414. To je zato ker je hitrost izračuna FP izrasa odvisna od dveh stvari: prvič od hitrosti prenosa operandov v in iz koprocesorja in od hitrosti same FP enote. S skrbnim uravnotežen jem teh faktorjev, postane eno samo integrirano vezja IMS T800-20 več kot petkrat hitrejše od kombinacije MC68000/MC68881 [7].

Procesor	Tip	Količina/s
Intel 80286/80287	8 MHz	300K
IMS T414-20	20 Mhz	663K
NS 32332-32081	15 Mhz	728K
MC68000/MC68881	16/12 Mhz Sun3	860K
VAX 11/780 FPA	Unix 4.3 BSD	1083K
IMS T800-20	20 Mhz	4000K
IMS T800-30	30 Mhz	6000K

Tabela 2
Primerjava procesorjev na osnovi Whetstonovega testa

Drugi pomembni kriteriji so še učinkovitost glede na površino silicija, zasedanje prostora na tiskanem vezju in potrebno št. dodatnih vezij. Poleg tega IMS T800 v mnogih aplikacijah ne potrebuje zunanjega pomnilnika, saj ga je na čipu že 4 Ksloge. Štirje IMS T800-30 zavzemajo enako površino na tiskanem vezju kot 80386 s WTL1167, obenem pa omogočajo šestkratno učinkovitost v vsaki sočasni aplikaciji.

Procesor	Količina/s
IMS T414-20	33K
Intel 80286/80287	37.5K
NS 32332-32081	48.5K
MC68000/MC68881	54K
IMS T800-20	200K
IMS T800-30	200K

Tabela 3
Normirana primerjava procesorjev na osnovi Whetstonovega testa pri taktu 1MHz

Tudi iz tabele 3 je razvidna moč T800 v primerjavi s predhodnikom T414. Ta moč izvira iz naslednjih dejstev:

- V testih T800 uporablja 4 Ksloge velik pomnilnik, ki je integriran v vezju.
- FPU je prav tako integrirana v vezje.
- FPU in CPU delujeta sočasno, pri čemer CPU računa naslove operandov v FP operacijah.

Rezultati bi bili za T800 manj ugodni, če bi naslavljal pomnilnik, ki ni na vezju: še slabši bi bili, če bi uporabljal serijske povezave za doseganje operandov.

6 ZAKLJUČEK

IMS T800 transputer je zelo zmogljiv gradnik za paralelne sisteme. Obenem dokazuje, da ni potrebno uporabljati koprocesorja, če želimo veliko numerično računalniško moč. Pri transputerju T800 najdemo na enem integriranem vezju centralno procesno enoto, numerično enoto za delo s števili v plavajoči vejici, pomnilnik in komunikacijski sistem. Skratka cel računalnik na enem vezju. Na primer: štiri Ksloge pomnilnika je v aplikacijah, kjer obdelujemo signale, ponavadi dovolj. Pri tem ne potrebujemo zunanjega pomnilnika. Zanimivo je tudi dejstvo, da je zmogljivost T800 pri računanja v plavajoči vejici večja od zmogljivosti mnogih drugih procesorjev pri računanju v številskih sistemih s fiksno piko.

To vezje bo osnova najmočnejšega evropskega superračunalnika, ki ga načrtujejo na univerzi v Edinburgu. Vsebovalo bo 1000 transputerjev in 1 Gslog pomnilnika. S takšno rešitvijo bo superračunalnik velik le slab kubičnih meter. Današnja tehnologija s katero je izdelan T800-20 ali T800-30 nudi 1.5 GFLOPS na 0.028 kubičnega metra strojne opreme.

Literatura

[1] Inmos reference manual for transputers, 1986
[2] Engineering data - IMS T414 transputer, 1986
[3] Engineering data - IMS T800 transputer, 1987
[4] Inmos product overwiew (transputer development system), 1986
[5] Inmos compilers writer's guide, 1987
[6] IMS T800 Architecture - technical note 6, 1988
[7] M Homewood, D May, D Shepherd, R Shepherd: The IMS T800 transputer: IEEE Micro, October 1987
[8] B Mihovilovič, S Mavrič, P Kolbezen: Transputer - osnovni gradnik večprocesorskih sistemov: Informatica 4, 1986
[9] F Mayer-Lindenberg: FIFTH on the transputer: Microprocessing and microprogramming, December 1987